JFP **18** (3): 421–422, 2008.
Printed in the United Kingdom

Book review

Logic in Computer Science: Modelling and Reasoning About Systems by Michael Huth and Mark Ryan, second edition. ISBN 0 521 54310 X.
doi:10.1017/S0956796807006715

Huth and Ryan's *Logic in Computer Science* is an upper-year undergraduate textbook on the subject and contains material on classical, modal and temporal logics, and their applications such as model checking, binary decision diagrams and program verification. These topics are of natural interest to a functional programmer. Specific prior knowledge in the field is no prerequisite but reading skills of formal mathematics and a maturity in mathematical thinking are clearly needed. Helping in improving the former, it makes a good job of introducing logical formalism in plain English.

The first chapter discusses propositional logic, and is a detailed and rigorous but accessible account on the topic. It starts with abstracting logical connectives from natural langague, and continues with natural deduction, a proof system for propositional logic. Next, logic is defined as a formal system, at the right point for an introductory textbook. Its semantics is treated with soundness and completeness theorems. Normal forms come next, including Horn clauses, and the chapter ends with SAT solvers. The explanations mix formulae and English text in a proper ratio for beginners.

The second chapter covers predicate logic in a similar format, also introducing Alloy, a tool for predicate logic model checking. The proof of undecidability of predicate logic makes a rather difficult reading here, but this makes the book serious. The third chapter on model checking introduces linear temporal logic and computational tree logic, also called branching-time logic, and the NuSMV model checker. The section concludes with a helpful discussion of the pros and cons of various temporal logics.

Program verification is treated in the fourth chapter, including Hoare triples, partial and total correctness with proof tableaux. Modal logic is presented in the fifth chapter, covering its syntax, semantics, together with 'logic engineering', an important topic; applications are also covered.

In the sixth chapter, a method of storing boolean functions is explained: the theory of Boolean Decision Diagrams, including ordered binary decision diagrams, which are a base for symbolic model checking that handles sets of states instead of individual ones, enabling much larger state spaces to be checked. The book ends with the relational μ-calculus.

The book does a good job of introducing logical formalism. Many texts provide terse definitions without explaining the why and how – Huth and Ryan carefully explain connectives, inference rules and metatheoretical properties, making their book accessible to a wide undergraduate audience.

There are some gems in the book, like the derivation of the universal quantifier introduction rule as a generalisation of the conjunction introduction rule. Another nice treatment is of the positive and negative aspects of proof theory and semantics, respectively; that is, proof theory can easily support a theorem and semantics can equally easily refute it. The introduction of material implication, a subject students tend to find it particularly challenging, is convincing, making the book suitable for beginners.

It is a fortunate feature of the book that the exercises are not just from mathematics and theoretical computer science but from practice, like the package management system modelled

with Alloy. Even though this is not a full-blown model, it shows that model checking is not just a scientists' toy. Another gem is the failed attempt to model mutual exclusion, which is described in Chapter 3. We know from practice that the perfect solution does not pop out from our heads for the first time. All through the book there are well-chosen examples and counter-examples of mathematical objects.

Huth and Ryan's textbook has few weaknesses, however. At first some general ones. The book unfortunately has a large number of exercises without solutions or, at least, hints. This would have been a serious work, and might have extended the number of pages, but without these, self-study of the text is hindered.

The reviewer should admit his bias towards mechanised theorem proving, so it is not a surprise that he would have liked to see it included in the book, along with λ-calculus, higher-order logic and other fields of logic in computer science. Of course, the authors have had to set a limit to the number of topics covered, as the field has a six-volume handbook – no textbook could cover all branches of this science. That said, the title of the book is not fortunate as it suggests a general coverage. A short mention of the Gődel incompleteness theorem would have made the book more complete.

In our age, when there are a number of mature functional languages, along with their implementations, it is surprising that there are no implementations of the algorithms in an executable functional language, only in imperative pseudo-code. To show this as an advantage to the readers of *JFP*, the reviewer thinks that the algorithms presented in the book are waiting to be implemented in a functional language.

There are also some more detailed comments to be made. In the first chapter, the book uses a sequential form of proofs, instead of proof trees. The reviewer would have preferred to see proof trees from the start, in line with the software engineering dictum that a notation should be easy to read, not to write.

Also, on page 44 the authors write, "Since every well-formed formula has finite height, we can show statements about all well-formed formulas by mathematical induction on their height. This trick is most often called structural induction, an important reasoning technique in computer science". I disagree here with the terminology, as this is mathematical, not structural induction. As the name suggests, structural induction uses the structure of the mathematical object to prove the properties of a whole class of it. Finally, presenting a finite version of the Knaster–Tarski theorem in the third chapter is not elegant, as the general proof is not that much longer and covers all cases.

At last, let me give you a summary on the book. Logic texts tend to be dry, terse, and full of formulae. Fortunately, this book did not follow this route. The book gives a thorough and clear introduction to the field encouraging students to read other books and it gives them an appetite to do so. The chapter on modal logic is especially well-written and stimulating. The book has a good style, with a preference given to knowledge over proofs, but details proofs when necessary. It is self-contained; reading it makes you familiar with the basics and applications of predicate, temporal and modal logics and of model checking. All things considered, I recommend buying it.

Gergely Buday
Budapest, Hungary

https://doi.org/10.1017/S0956796807006715 Published online by Cambridge University Press